Systèmes embarqués

Capteurs, actuateurs et programmation des microcontrôleurs

Pape Abdoulaye BARRO, PhD
UFR des Sciences et technologies
Département Informatique

13 août 2022

Contenu

Contenu

Capteurs : Définition

Définition 1.1.

L'organe permettant d'élaborer, à partir d'une grandeur physique observée (température, pression, position, concentration, etc.), une grandeur physique utilisable (souvent électrique) à des fins de mesure ou de commande.

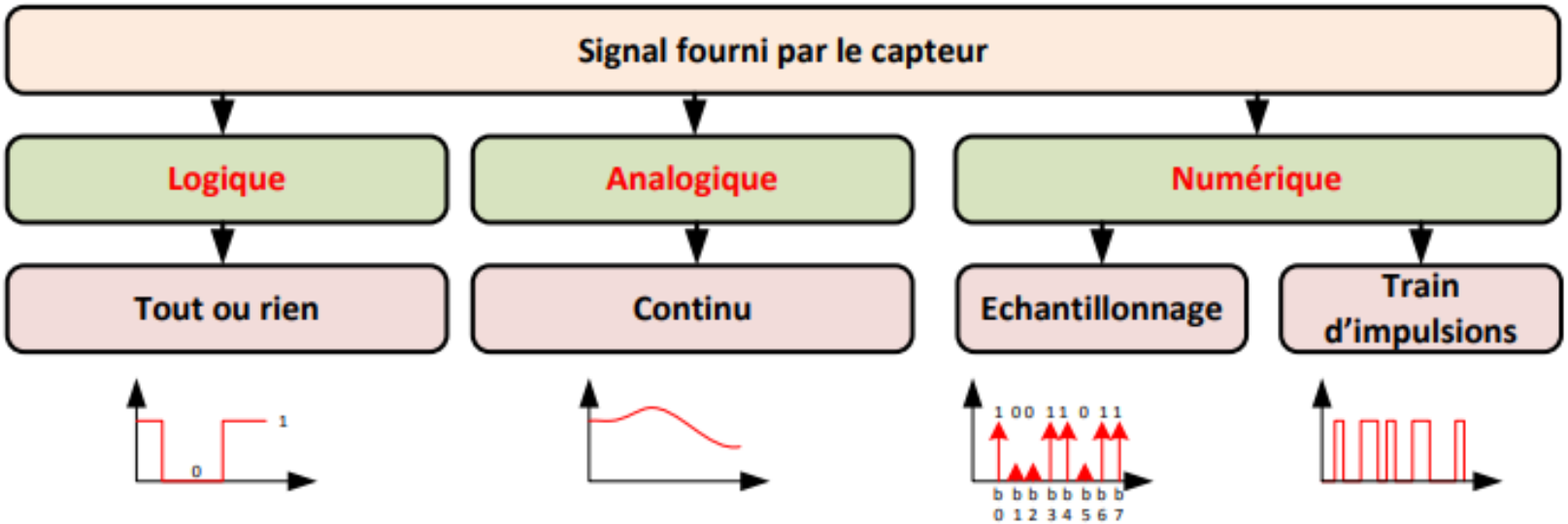
- Ils sont donc à la base des systèmes d'acquisition de données et leur mise en œuvre est du domaine de l'instrumentation ;
- Un capteur est caractérisé soit par **son temps de réponse**, **sa linéarité**, **la grandeur physique observée**, **sa gamme de mesure**, **sa précision**, **sa sensibilité**, **sa résolution**, etc ... .

Capteurs : Classification 1

- Ils sont classés en types de capteurs, à savoir les capteurs actifs, les capteurs passifs, etc ... .
  - **Capteurs actifs :** qui ont la capacité de convertir en énergie électrique la forme d'énergie de la mesurande : thermique, mécanique, ..., en utilisant des effets tels que l'induction électromagnétique, Hall, piézoélectrique, thermoélectrique, pyroélectrique, photoélectrique, Faraday, ... Ils sont présentés comme un générateur car ils délivrent à leur sortie soit une tension, soit un courant, soit une charge électrique ;
  - **Capteurs passifs :** qui sont une sorte d'impédance sensible à la mesure. Ils peuvent délivrer une grandeur telle que la variation de l'impédance, la résistance, l'inductance ou la capacité et il est donc nécessaire de leur appliquer une tension pour obtenir un signal de sortie ;
  - **...** .

Capteurs : Classification 2

- Les capteurs peuvent être classés en trois groupes selon la nature de l'information de sortie :



Capteurs : Quelques capteurs disponibles I

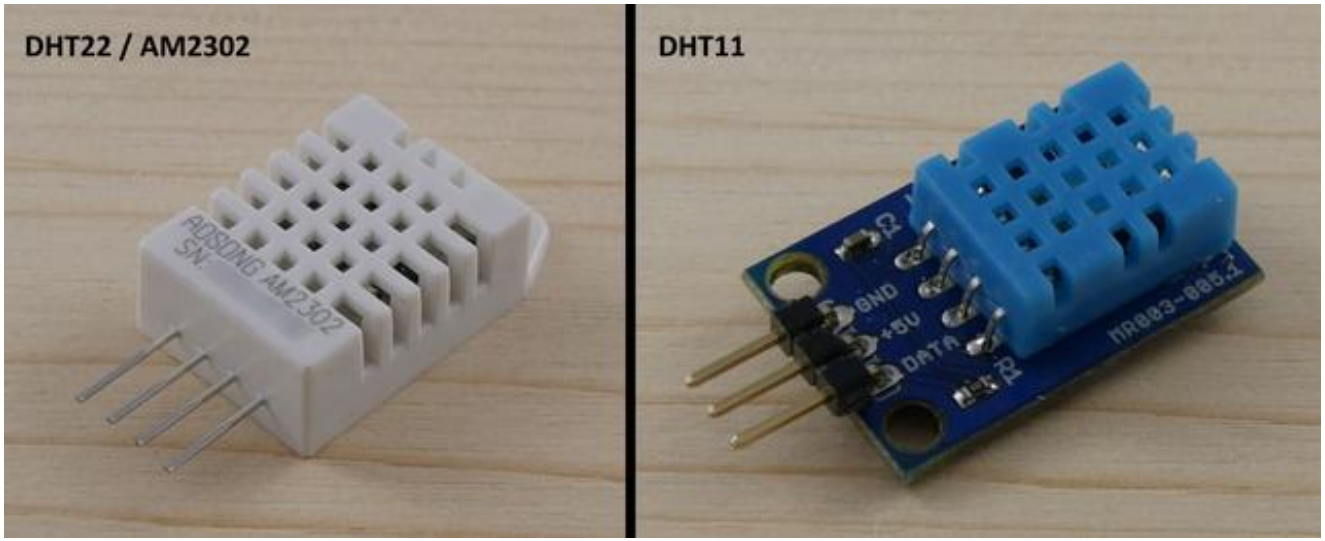
Voici, ci-dessous, quelques-uns des capteurs disponibles et accessibles sur le marché :

- **Capteurs de température analogique :** LM35, TMP36, ... ;
- **Capteurs de température numérique :** DS18 ;
- **Capteurs de température et d'humidité :** DHT11, DHT22, SHT11, SHT15, SHT21, ... ;
- **capteur de luminosité :** Photorésistance, capteur infrarouge (pour éviter les obstacles), capteur de détection de rayons ultraviolets, capteur de couleur ;
- **Capteur de distance :** Capteurs à ultrasons (HC-SR04) ;
- **Capteur de mouvement :** HC-SR501 ;
- **Capteur de gaz et de fumée :** les MQ (MQ2, MQ3, ...) ;
- **Capteur de son analogique :** KY-038 ;
- **Capteur de vibration numerique :** SW-420 ;
- **Capteur ou lecteur RFID :** RC522 ;
- **Capteur ou lecteur d'empreintes digitales :** AS608 ;
- **Capteur ou détecteur d'incendie :** KY-026 ;

Exemple de Capteur : DHT11, DHT12

Le DHT22 et le DHT11 sont des capteurs de température et d'humidité.

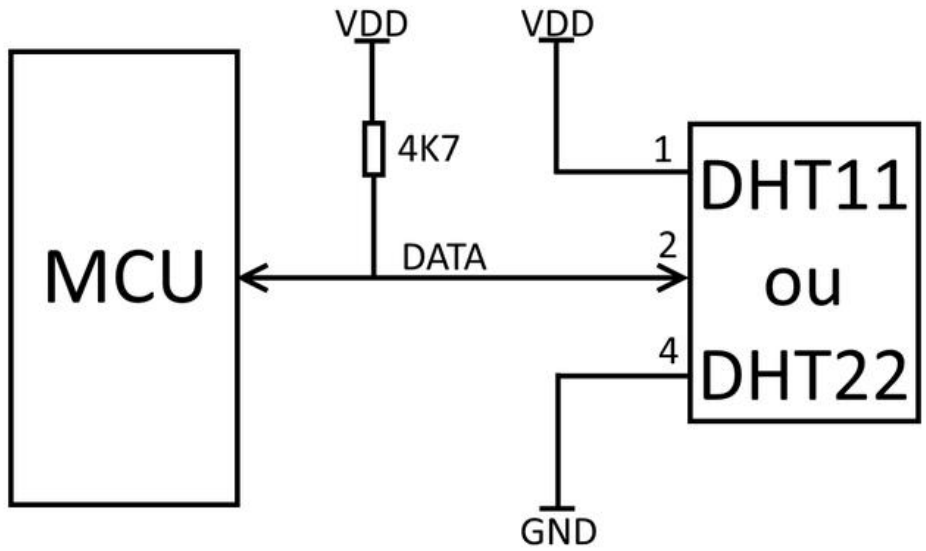
- Le capteur DHT22 / AM2302 est capable de mesurer des températures de -40 à +125°C avec une précision de +/- 0,5°C et une humidité relative de 0 à 100% avec une précision de +/- 2% (+/- 5% aux extrêmes, 10% et 90%). Une mesure peut être effectuée toutes les 500 millisecondes.



Exemple de Capteur : DHT11, DHT12 (Suite)

...

- Le capteur DHT11 est capable de mesurer des températures de 0 à +50°C avec une précision de +/- 2°C et une humidité relative de 20 à 80% avec une précision de +/- 5%. Une mesure peut être effectuée toutes les secondes.
- ils consomment 3,3 volts ou 5 volts et ont le même câblage et protocole de communication.
- leur schéma de câblage est :

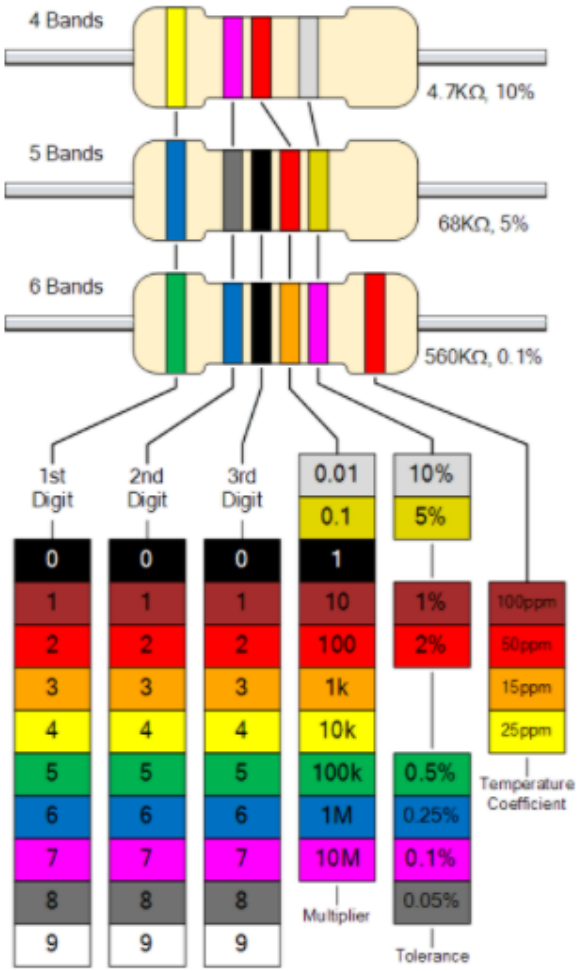


Capteurs (commande) : Les Boutons-poussoirs

Dispositif de commande manuelle permettant de contrôler un aspect d'une machine ou d'un processus (par exemple, allumer/éteindre une lampe, etc.).

Capteurs (protection) : code couleur des résistances

Il est très utile de connaître la valeur d'une résistance, bien que la plupart des modules comprennent maintenant les composants nécessaires à l'interfaçage avec les microcontrôleurs.



Contenu

Actionneur : Définition I

Définition 2.1.

Un actionneur est un composant d'une machine qui permet d'effectuer des mouvements physiques en convertissant l'énergie (électrique, pneumatique, hydraulique, ...) en force mécanique (mouvement linéaire (poussée/traction) ou rotatif).

- Les actionneurs peuvent être dans un système de commande simple, tel qu'un moteur mécanique, ou dans un système informatique, tel qu'un robot.

Actionneur : les types d'actionneur I

Les actionneurs peuvent être classés en fonction du type d'énergie qu'ils utilisent et du type de mouvement qu'ils produisent.

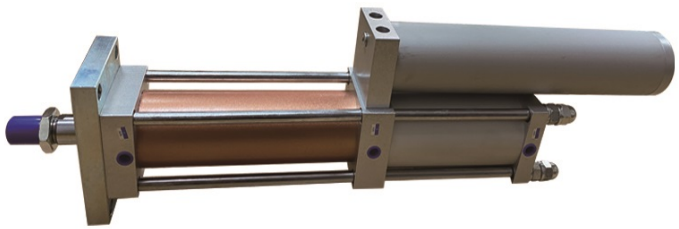
- **Les actionneurs électriques :** ce sont les plus courants. Ils convertissent l'énergie électrique du courant continu ou alternatif en énergie mécanique. On distingue les actionneurs électriques linéaires et les actionneurs électriques rotatifs (les moteur en mvt continu, les servomoteurs et les moteurs pas à pas).

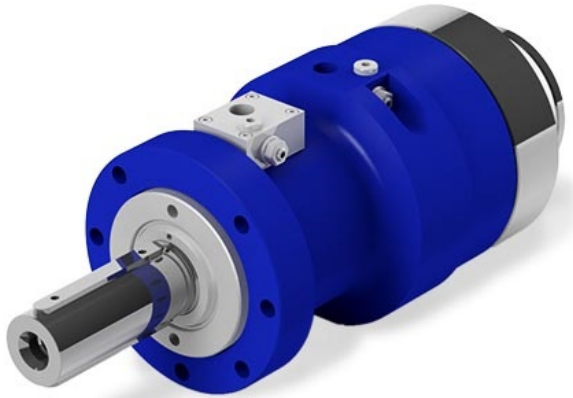
Actionneur : les types d'actionneur II

- **Les actionneurs pneumatiques :** Les actionneurs pneumatiques permettent d'effectuer de grands mouvements linéaires ou rotatifs à basse pression en utilisant de l'air comprimé. Leur force réside dans leur mouvement rapide, point à point, et ils ne sont pas facilement endommagés par les butées..

Actionneur : les types d'actionneur III

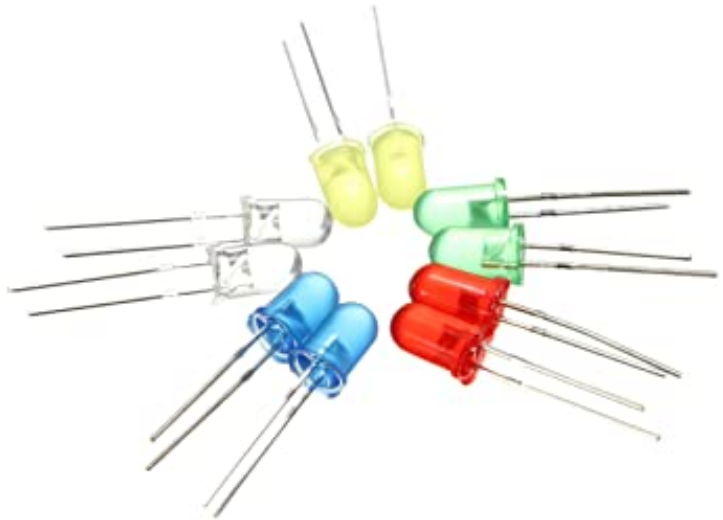
- **Les actionneurs hydraulique :** Les actionneurs hydrauliques utilisent l'énergie hydraulique pour créer un mouvement linéaire ou rotatif et sont très puissants en raison de la quasi incompressibilité des liquides.





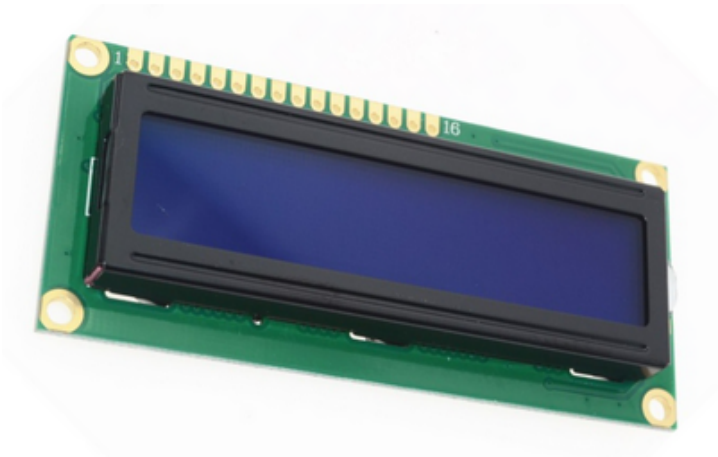
Actionneur : Les ampoules et les leds

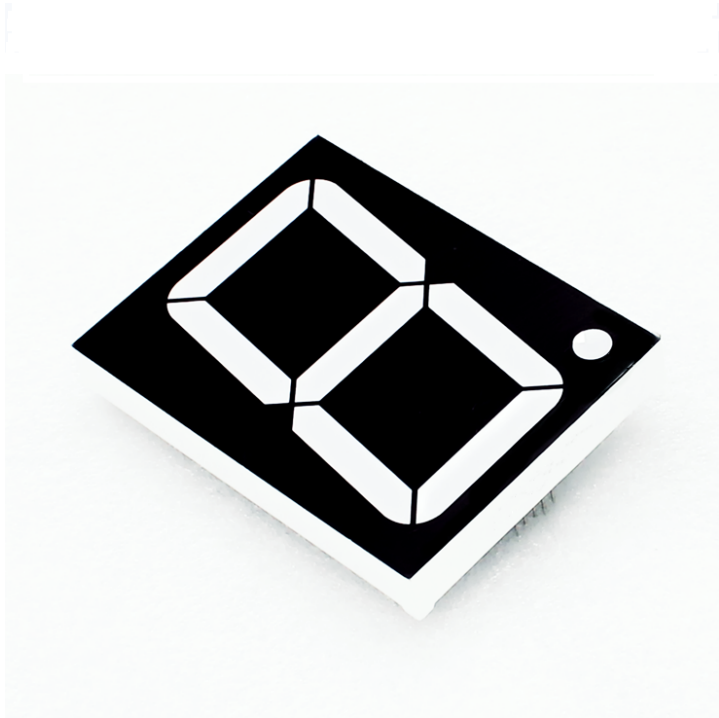
Les **ampoules** et les **leds** peuvent être mis dans cette catégorie. Elles ont la capacité de transformer l'énergie électrique en énergie lumineuse.



Actionneur : Les ecrans LCD et les afficheurs 7 segments

Les **ecrans LCD** (Liquid Crystal Display) et les **afficheurs 7 segments** sont également largement utilisés dans le domaine des systèmes embarqués (montres, tableau de bord, calculatrices, etc.) en raison de leur faible consommation d'énergie et de leur faible coût d'acquisition.





Contenu

Premiers pas avec Arduino

Arduino est un microcontrôleur open-source qui permet la programmation et l'interaction. Il est basé sur C/C++ avec une bibliothèque Arduino pour lui permettre d'accéder au matériel.

Jusqu'à présent, grâce à sa nature open-source, les utilisateurs ont construit des cartes Arduino de différentes tailles, formes et niveaux de puissance pour contrôler leurs projets.

Arduino est composé de deux parties principales :

1. La carte Arduino, qui est le matériel sur lequel vous travaillez lorsque vous construisez vos objets ;
2. L'IDE Arduino, qui est le logiciel que vous exécutez sur votre ordinateur pour créer un croquis (un petit programme informatique) que vous téléchargez sur la carte Arduino.

Variantes Arduino

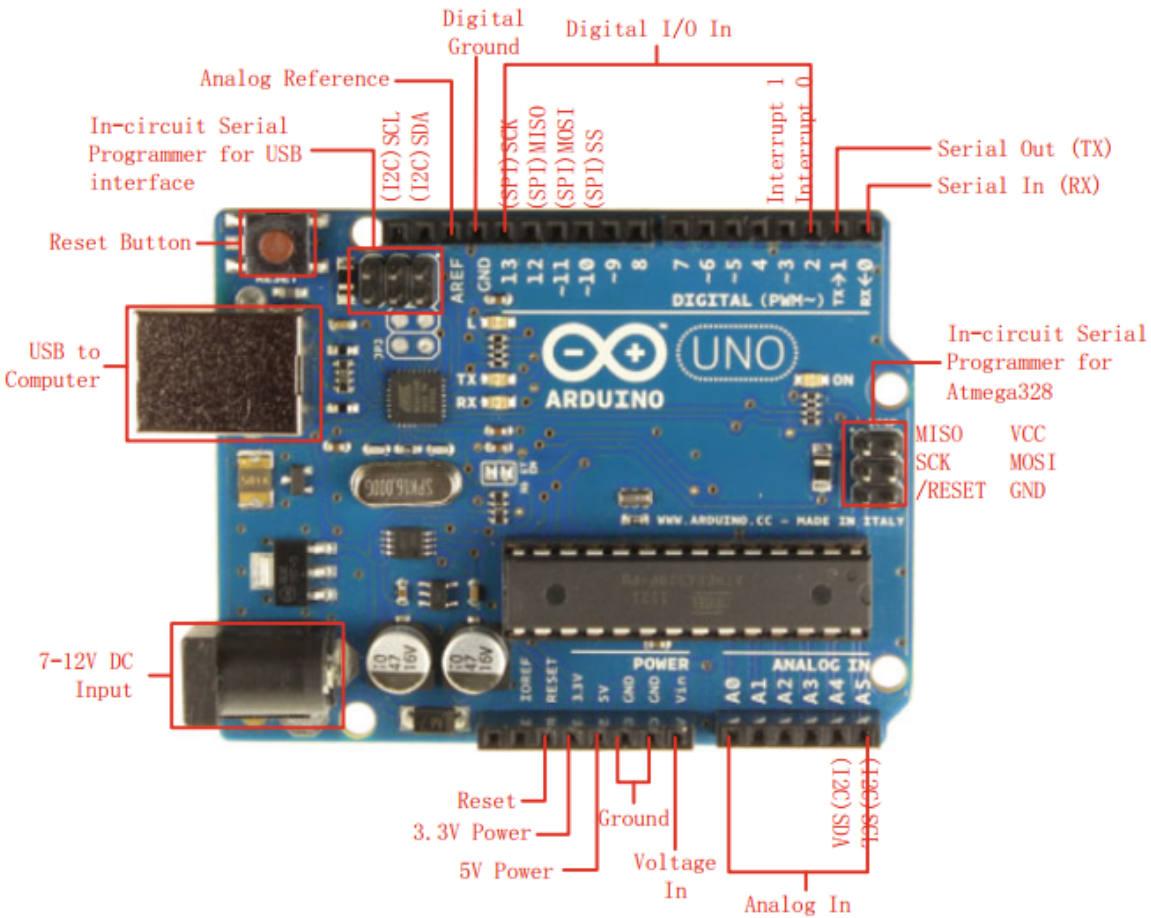
Il existe de nombreux types de microcontrôleurs Arduino qui diffèrent non seulement par leur conception et leur fonctionnalité, mais aussi par leur taille et leurs capacités de traitement. Cependant, seuls deux modèles utilisent des puces complètement différentes : le modèle Standard utilise la puce **Atmega 8/168/328** et le modèle Mega utilise la puce **Atmega1280**, plus robuste, avec plus de broches d'entrée/sortie.

Il existe de nombreux autres fabricants qui utilisent des schémas open-source fournis par Arduino pour fabriquer leurs propres planches (soit identiques à l'original, soit avec des variations pour ajouter des fonctionnalités), par exemple, **DFRobot**.

Dans ce cours, nous utilisons uniquement la carte Arduino Uno R3.

Arduino Uno R3

Arduino Uno est une carte à microcontrôleur basée sur l'ATmega328P. Elle contient tout le nécessaire pour supporter le microcontrôleur.

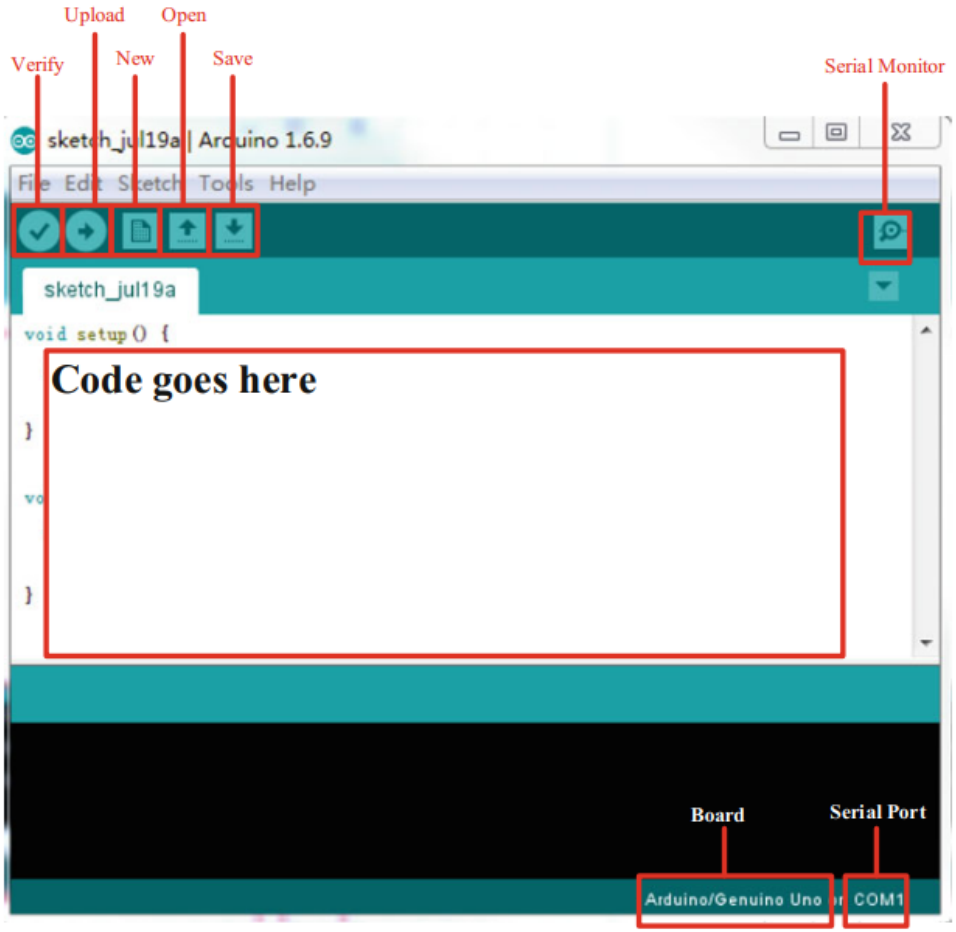


Elle est essentiellement constituée de :

- 14 broches d'entrée/sortie numériques ;
- 6 entrées analogiques ;
- un résonateur céramique de 16 MHz ;
- une connexion USB ;
- une prise d'alimentation ;
- un bouton de réinitialisation ;
- etc.

Installez les pilotes

Avant de commencer votre travail, vous devez d'abord télécharger l'environnement de développement (l'IDE) sur : www.arduino.cc/en/Main/Software.



Les fonctions de base I

Le langage Arduino est basé sur C/C++ et supporte toutes les constructions C standard ainsi que certaines fonctionnalités C++. Il est lié à **AVR Libc** et permet l'utilisation de n'importe laquelle de ses fonctions.

Structures	
void setup()	La fonction est appelée quand un sketch commence. Utilisez-la pour initialiser des variables, les pin modes, commencer à utiliser des bibliothèques, etc. La fonction de configuration ne sera exécutée qu'une seule fois, après chaque mise sous tension ou réinitialisation de la carte Arduino.
void loop()	la fonction fait des boucles consécutives, permettant à votre programme de changer et de répondre. Utilisez-le pour contrôler activement la carte Arduino.

Les fonctions de base II

Structures de contrôle	
if (condition) {}	Teste si une certaine condition a été atteinte. Utilisé en conjonction avec un opérateur de comparaison.
if. . . . . else	Permet d'effectuer plusieurs tests.
for(initialization ; condition ; increment)	Crée une boucle pour les opérations répétitives.
switch (var) { **case label : break ; default : break ;** }	Vous permet de spécifier différents codes qui doivent être exécutés dans diverses conditions.
while()	Boucles continues, et infinies, jusqu'à ce que l'expression à l'intérieur de la parenthèse, devienne fausse.
do {} **while ()**	Fonctionne de la même manière que la boucle while, à l'exception du fait que l'état est testé à la fin de la boucle, tout comme la boucle do, qui sera toujours exécutée au moins une fois.

Les fonctions de base III

Suite ...

Structures de contrôle	
break	Sorties d'une boucle do, for ou while, en contournant la condition normale de la boucle.
continue	Saute le reste de l'itération actuelle d'une boucle (do, for, ou while). Il continue en vérifiant l'expression conditionnelle de la boucle, et procède à toutes les itérations suivantes.
return	Termine une fonction et renvoie une valeur d'une fonction à la fonction appelante.
goto	Transfère le flux du programme vers un point labellisé dans le programme.

Les fonctions de base IV

Suite ...

Autre syntaxe	
;	Chaque déclaration se termine par un point-virgule.
{ }	Les accolades bouclées viennent toujours par paires ; ils sont utilisés pour définir le début et la fin des fonctions, des boucles et des instructions conditionnelles.
//	Commentaire d'une seule ligne.
/* */	Commentaire de plusieurs lignes.
#define	Utilisé pour donner un nom à une valeur constante.
#include	Inclure les bibliothèques extérieures dans votre croquis.

Les fonctions de base V

Suite ...

Opérateurs de comparaison	
x == y	x est égal à y.
x != y	x n'est pas égal à y.
$x < y$	x est inférieur à y.
$x > y$	x est supérieur à y.
$x <= y$	x est inférieur ou égal à y.
$x >= y$	x est supérieur ou égal à y

Les opérateurs booléens	
&&	ET logique. Vrai seulement si les deux opérandes sont vrais.
||	OU logique. Vrai si l'un des opérandes est vrai.
!	NON logique. Vrai si l'opérande est faux

Les fonctions de base VI

Suite ...

Constantes	
HIGH / LOW	Lors de la lecture ou de l'écriture sur une broche numérique, il n'y a que deux valeurs possibles qu'une broche peut prendre (ou être réglée sur) : HIGH et LOW
true / false	Niveaux logiques (résultat d'une comparaison) : false est défini comme 0, true est défini comme 1 (mais plus largement, tout sauf 0).
INPUT / OUTPUT	Les broches numériques peuvent être utilisées en INPUT (entrée) ou en OUTPUT (sortie). Changer une broche de INPUT à OUTPUT avec pinMode() change radicalement le comportement électrique de la broche.

Les fonctions de base VII

Suite ...

Types de données	
void	Utilisé dans les déclarations de fonction pour indiquer que la fonction ne renvoie aucune information.
boolean	Un booléen possède l'une des deux valeurs suivantes, vraie ou fausse.
char	Un type de données qui stocke une valeur de caractère.
unsigned char	Un type de données non signé qui occupe 1 octet de mémoire. Identique au type de données byte. Le type de données char non signé encode les nombres de 0 à 255.
byte	Un byte stocke un nombre non signé de 8 bits, de 0 à 255.
int	Les nombres entiers sont votre principal type de données pour le stockage des nombres, et stockent une valeur de 2 octets. Cela donne une plage de -32 768 à 32 767 .
unsigned int	Les ints non signés (entiers non signés) sont identiques aux **int** en ce sens qu'ils stockent une valeur de 2 octets. Au lieu de stocker des nombres négatifs, ils ne stockent que des valeurs positives, ce qui donne une plage utile de 0 à 65 535 .

Les fonctions de base VIII

Types de données	
word	Un word stocke un nombre non signé de 16 bits, de 0 à 65536. Même chose qu'un int non signé.
long	Les variables de type long sont des variables de taille étendue pour le stockage des nombres et stockent 32 bits (4 octets), de -2 147 483 648 à 2 147 483 647.
unsigned long	Contrairement aux longs standards, les longs non signés (unsigned long) ne stockent pas de nombres négatifs, ce qui fait que leur plage de stockage va de 0 à 4 294 967 295.
float	Type de données pour les nombres à virgule flottante, un nombre qui a un point décimal.
double	Les nombres à virgule flottante sont souvent utilisés pour approximer des valeurs analogiques et continues car ils ont une plus grande résolution que les nombres entiers. Les nombres à virgule flottante ont une précision de 6 à 7 chiffres décimaux.
string	Les chaînes sont représentées sous forme de tableaux de type char et se terminent par null.
arrays	Il est souvent pratique, lorsqu'on travaille avec de grandes quantités de texte, comme dans le cas d'un projet avec un écran LCD, de mettre en place une série de chaînes

Les fonctions de base IX

Suite ...

Conversion	
char()	Convertir une valeur en char.
byte()	Convertir une valeur en byte.
int()	Convertir une valeur en int.
word()	Convertir une valeur en word ou créer un mot à partir de deux octets.
long()	Convertir une valeur en long.
float()	Convertir une valeur en float.
Utilitaires	
sizeof	L'opérateur sizeof renvoie le nombre d'octets d'un type de variable ou le nombre d'octets occupés par un tableau.

Les fonctions de base X

Suite ...

Fonctions : E/S numériques	
pinMode(pin, mode)	Configure la broche spécifiée pour qu'elle se comporte comme une entrée ou une sortie. pin est le numéro de pin.
digitalWrite(pin, value)	Ecrire une valeur HIGH ou LOW sur une broche numérique.
digitalRead(pin)	Lire la valeur à partir d'une broche numérique spécifiée. Le résultat sera soit HIGH, soit LOW.
Fonctions : E/S analogiques	
analogReference(type)	La tension de référence par défaut est de 5 V.Cela peut être changé en un autre type et une résolution différente en utilisant cette fonction.
analogRead(pin)	Lire la valeur de la broche analogique spécifiée et renvoie une valeur comprise entre 0 et 1023 pour représenter une tension entre 0 et 5 V (par défaut). Il faut environ 0,0001 s pour lire une broche analogique.
analogWrite(pin,value)	Ecrire une valeur analogique (onde PWM) sur une broche. valeur est le rapport cyclique : entre 0 (toujours désactivé) et 255 (toujours activé). Fonctionne sur les broches 3, 5, 6, 9, 10 et 11.

Les fonctions de base XI

Suite ...

Fonctions : communication série	
Serial.begin(9600)	Utilisé pour commencer les communications série, généralement à un débit de 9600 bauds (bits par seconde).
Serial.print(val,format)	Imprime les données sur le port série sous forme de texte ASCII lisible par l'homme.
Serial.println(val)	Imprime le val suivi du retour chariot.
Serial.available()	Obtenez le nombre d'octets (caractères) disponibles pour la lecture à partir du port série. Il s'agit des données déjà arrivées et stockées dans le tampon de réception série (qui contient 128 octets).
Serial.read()	Lire les données série entrantes.
Serial.write()	Écrire des données binaires sur le port série.
Serial.end()	Désactive la communication série, ce qui permet d'utiliser les broches RX et TX pour les entrées et sorties générales.

Les fonctions de base XII

Suite ...

Temps	
delay(ms)	Suspend le programme pendant la durée (en millisecondes) spécifiée en paramètre.
delayMicroseconds(us)	Met le programme en pause pendant la durée (en microsecondes) spécifiée en paramètre.
micros()	Retourne le nombre de microsecondes depuis que la carte Arduino a commencé à faire fonctionner le programme actuel.
millis()	Renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le programme en cours.

Les fonctions de base XIII

Suite ...

Fonctions : mathématiques	
min(x, y)	Calcule le minimum de deux nombres.
max(x, y)	Calcule le maximum de deux nombres.
abs(x)	Calcule la valeur absolue d'un nombre.
pow(base, exponent)	Calcule la valeur d'un nombre élevé à une puissance.
sqrt(x)	Calcule la racine carrée d'un nombre.
map(value, fromLow, fromHigh, toLow, toHigh)	Remappe un nombre d'une plage à une autre. Autrement dit, une valeur de fromLow serait mappée à toLow, une valeur de fromHigh à toHigh, des valeurs intermédiaires à des valeurs intermédiaires.

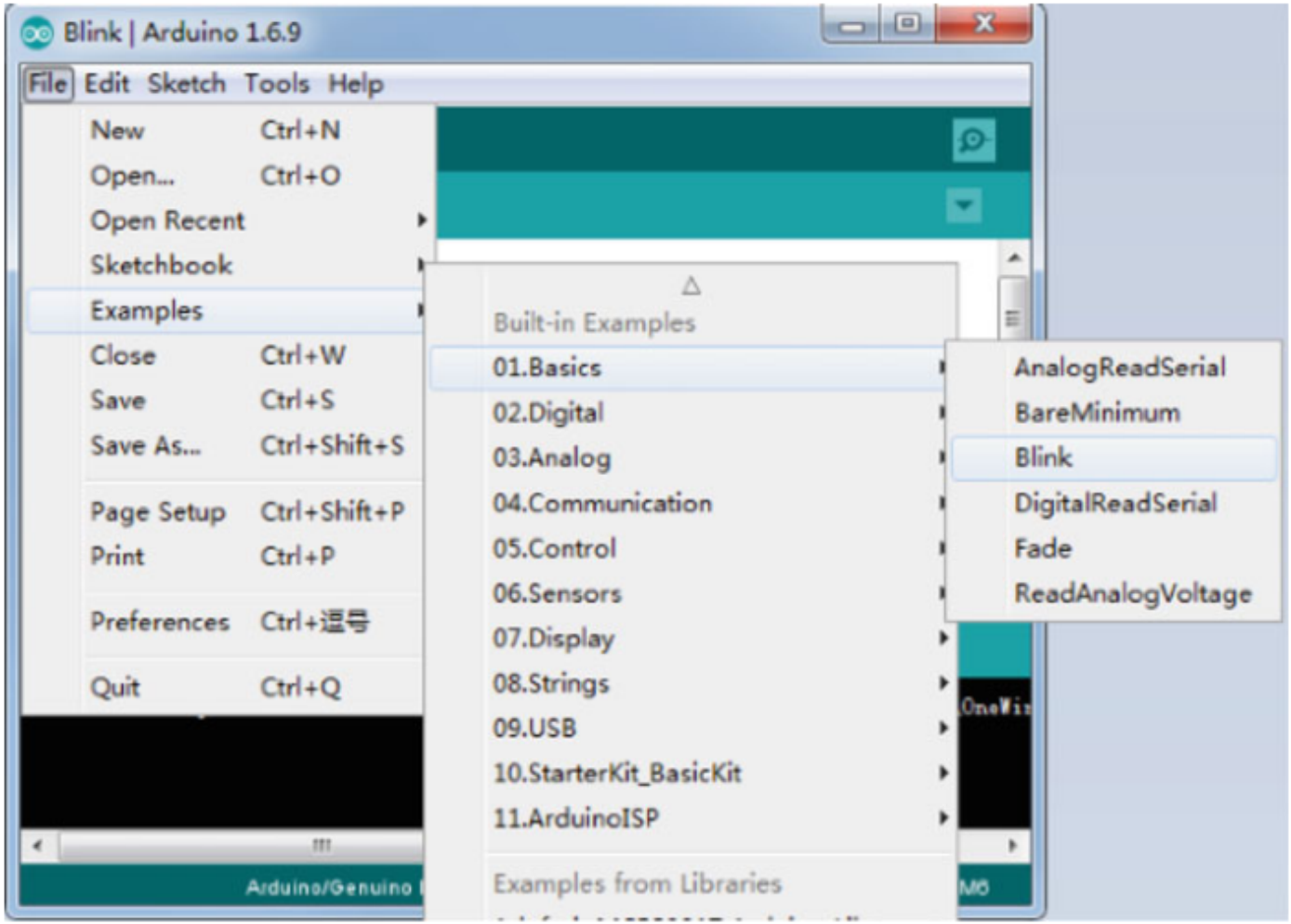
Les fonctions de base XIV

Suite ...

Fonctions : trigonométrique	
sin()	Calcule le sinus d'un angle (en radians). Le résultat sera compris entre -1 et 1.
cos()	Calcule le cos d'un angle (en radians). Le résultat sera compris entre -1 et 1.
tan()	Calcule la tangente d'un angle (en radians). Le résultat sera compris entre l'infini négatif et l'infini.
Fonctions : nombres aléatoires	
randomSeed(seed)	Initialise le générateur de nombres pseudo-aléatoires, le faisant démarrer à un point arbitraire de sa séquence aléatoire.
random()	La fonction aléatoire génère des nombres pseudo-aléatoires.

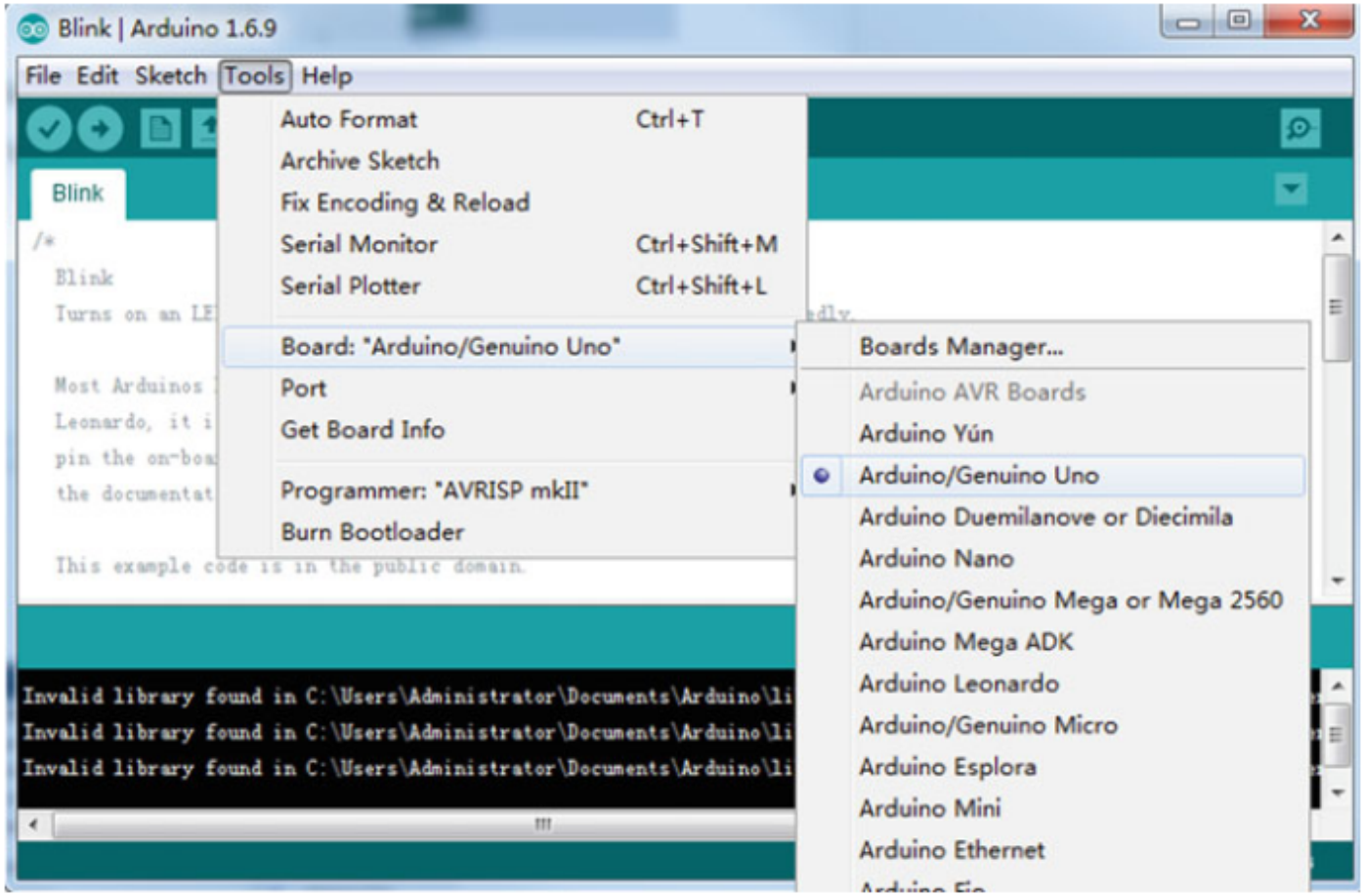
“Blink”

”Blink” dans IDE : Modifiez l’exemple de Blink pour allumer/éteindre une led ;



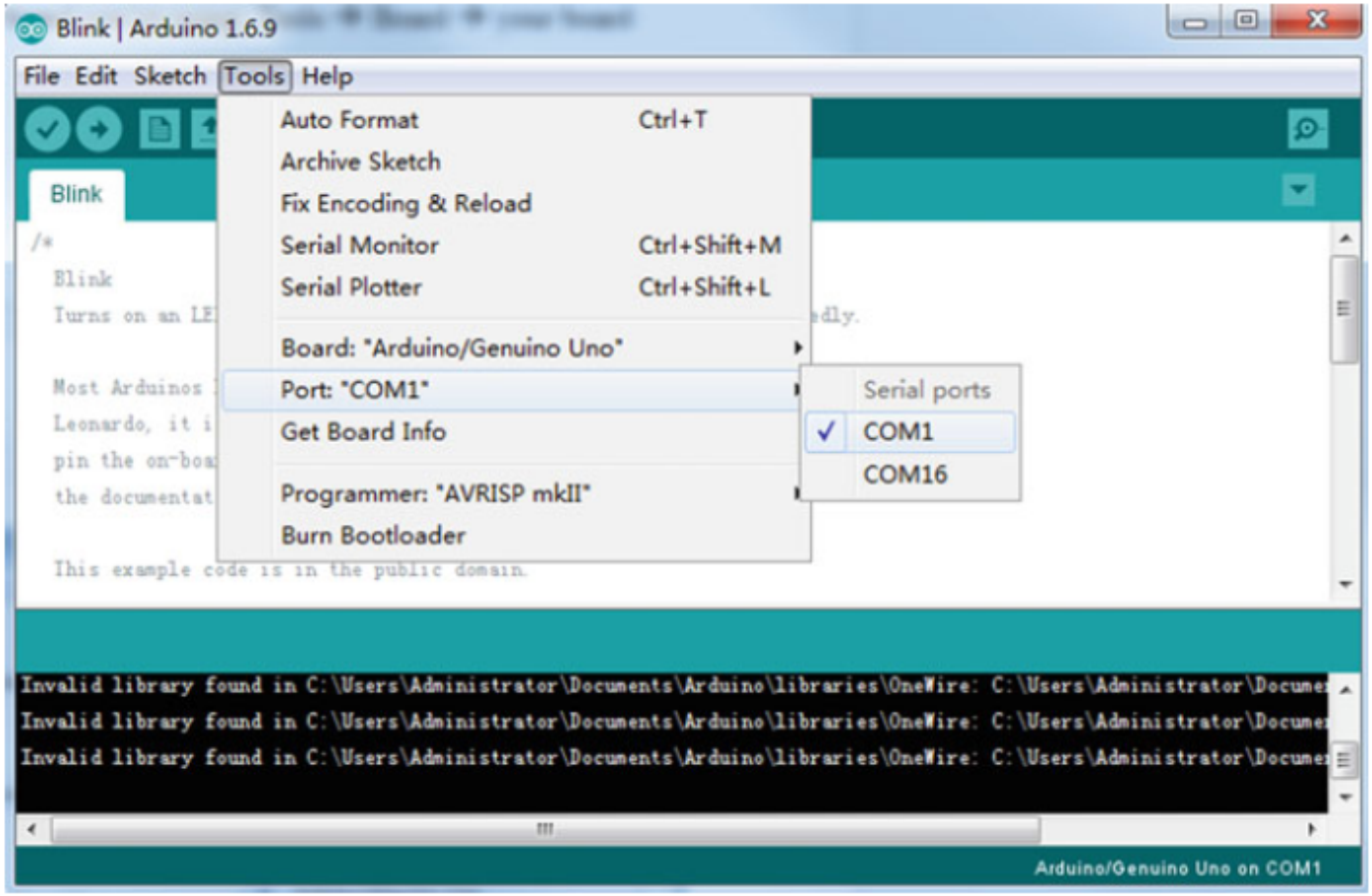
“Board”

Sélection du type de carte Arduino dans l’IDE.



“Port”

Sélection du port série .



Travaux Pratiques

l'apprentissage pratique ...

Thanks

Systèmes embarqués

Capteurs, actuateurs et programmation des microcontrôleurs

Pape Abdoulaye BARRO, PhD
UFR des Sciences et technologies
Département Informatique

13 août 2022